PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	Christopher Peiffer	Confirmation No.	9849
Serial No.:	09/975,286		
Filed:	October 10, 2001	Customer No.:	72689
Examiner:	Haresh N. Patel		
Group Art Unit:	2154		
Docket No.:	1014-152US01/JNP-0489		
Title:	STRING MATCHING METHOD AND DEVICE		

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450,
Alexandria, VA 22313

Dear Sir:

This is an Appeal Brief responsive to the Final Office Action mailed February 22, 2007. The Notice of Appeal was filed on May 22, 2007. Please also charge any additional fees that may be required or credit any overpayment to Deposit Account No. 50-1778.

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest is Juniper Networks, Incorporated, of Sunnyvale, California.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1-5, 7, 8, and 11-26 are on appeal in this case.

Claims 1 and 24-26 stand rejected under 35 U.S.C. 112, first paragraph.

Claims 1-5, 7, 8, 11-20, 22 and 23 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,842,680 to Brandstad in view of HTTP 1.1, Fielding et al., June 28, 2001, pages 1-6, Chapter 3 Protocol Parameters", pages 1-10, Chapter 4 HTTP Message, pages 1-4 ("Fielding") and U.S. Patent 6,377,991 to Smith et al. (Hereinafter "Smith").

Claim 21 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Brandstad, Fielding, Smith, and "Official Notice" in view of Slater et al. (Hereinafter "Slater").

Claims 24-26 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Brandstad, Fielding, Smith, and "Official Notice" in view of U.S. Patent 6,523,108 to James et al. (Hereinafter "James").

STATUS OF AMENDMENTS

No amendments have been filed subsequent to the Final Rejection mailed February 22, 2007, from which this Appeal has been made.

SUMMARY OF CLAIMED SUBJECT MATTER

Claim 1

Claim 1 recites a computer-implemented method for comparing an unknown string to a predefined string. Claim 1 requires storing, on a network device, a database

containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers; the predefined string headers may be stored in memory and/or accessed from a linked database on the network device.

Claim 1 requires receiving, with the network device, a network message. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 of the network device 18 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 1 requires, in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102, e.g., a predefined HTTP header.

Claim 1 requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes identifying the unknown string at 102 that is described as typically a header of the received message to be compared with the predefined string selected from the database.

Claim 1 requires performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. Fig. 6 and pg. 12, ll. 1-4 further describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the unknown header and the predefined string. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation between segments of the unknown header and segments of the predefined string.

Claim 1 requires performing a bitwise operation between a predefined flag and a result of the exclusive OR operation. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result

of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 1 requires comparing the predefined flag and a result of the bitwise operation to produce an indication for a case-insensitive string match, wherein the indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description states that the comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application on pg. 13 states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 1 requires processing the network message based on the indication of the case-insensitive string match. Claim 1 also requires outputting a response from the network device based on the processed network message. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output.

Claim 24

Claim 24 recites a method of case-insensitive string matching for use in a computer network. Claim 24 requires storing, on a network device, a plurality of predefined strings, wherein the predefined strings represent known headers for a network communication protocol. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers, and may be stored in memory and/or accessed from a linked database.

Claim 24 requires receiving, with the network device, a network message. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 24 requires selecting one of the plurality of predefined strings stored within the network device. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102, the predefined string being described as an HTTP header.

Claim 24 requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes identifying an unknown string at 102 that is described as typically a message header to be compared with the predefined string.

Claim 24 requires performing at least one bitwise exclusive OR operation between characters of the selected predefined string and corresponding characters of the unknown string. Fig. 6 and pg. 12, ll. 1-4 describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the strings. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation on segments of each of the strings.

Claim 24 requires performing a bitwise OR operation between a result of the bitwise exclusive OR operation and a predetermined flag. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 24 requires comparing the predetermined flag and a result of the bitwise OR operation to produce a single bit output that indicates whether a case-insensitive match exists between the selected predefined string and the unknown string. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description describes that comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 24 requires processing the network message based on the indication of the case-insensitive match. Claim 24 also requires outputting a response from the network device. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output.

Claim 25

Claim 25 recites a computer networking device for improving data transfer via a computer network, the device comprising a processor configured to compare a client HTTP header with a known HTTP header. Claim 25 requires storing, on the networking device, a database containing a plurality of known HTTP headers. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers, and may be stored in memory and/or accessed from a linked database. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 25 requires receiving, with the networking device, a client HTTP header and, in response to receiving the client HTTP header, selecting one of the known HTTP headers stored within the database of the network device. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102 that is typically an HTTP header.

Claim 25 requires performing a bitwise exclusive OR operation on binary representations of the client HTTP header and the known HTTP header selected from the database. Fig. 6 and pg. 12, ll. 1-4 describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the strings. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation on segments of each of the strings.

Claim 25 requires performing a bitwise OR operation between a result of the exclusive OR operation and a predetermined flag. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 25 requires comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for a case-insensitive string match between the client HTTP header and the selected known HTTP header. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description describes that comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 25 requires processing the network message based on the indication of the case-insensitive match. Claim 25 also requires outputting a response from the network device based on the processed network message. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output

Claim 26

Claim 26 recites an article of manufacture comprising a storage medium having a plurality of machine-readable instructions. Claim 26 requires storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a

string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers, and may be stored in memory and/or accessed from a linked database.

Claim 26 requires receiving, with the network device, a network message and, in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 26 requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102 that is typically an HTTP header.

Claim 26 requires performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes identifying the unknown string at 102 that is described as typically a message header to be compared with the predefined string.

Claim 26 requires performing a bitwise operation between a predefined flag and a result of the exclusive OR operation. Fig. 6 and pg. 12, ll. 1-4 describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the strings. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation on segments of each of the strings. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 26 requires comparing the predefined flag and a result of the bitwise OR operation to produce an indication for a case-insensitive string match between the predefined string and the unknown string, wherein the indication for the case-insensitive match indicates whether all characters of the unknown string within the network message match all corresponding characters of the identified predefined string so as to match one

of the known headers of the network communication protocol. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description describes that comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 26 requires processing the network message based on the indication of the case-insensitive match; and outputting a response from the network device based on the processed network message. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Appellant submits the following grounds of rejection to be reviewed on Appeal:

(1) The first ground of rejection to be reviewed is the rejection of claims 1 and 24-26 stand rejected under 35 U.S.C. 112, first paragraph.

(2) The second ground of rejection to be reviewed is the rejection of claims 1-5, 7, 8, 11-20, 22 and 23 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Brandstad in view of Fielding and Smith et al. (Hereinafter "Smith") and "Official Notice."

ARGUMENT

Appellant respectfully traverses the current rejections advanced by the Examiner, and requests reversal by the Board of Patent Appeals based on the arguments below. The applied references fail to disclose or suggest features of Appellant's claims.

The First Ground of Rejection to be Reviewed on Appeal

The Examiner rejected claims 1, 24, and 25 under 35 U.S.C. 112, first paragraph as not including elements that are critical or essential to practice the invention. (Final Office Action, Feb. 22, 2007, pg. 3). Specifically, the Examiner stated:

> *Without using the storage such as memory it is not possible to store/retain/hold the database. Without using the some [sic] network and some network interface card it is not possible to receive the network message. Without using the computer (and computer network, claim 25, and computing system claim 26) it is not possible to process the network message. Without using some network interface card it is not possible to output the response from the network device. Further, it is not possible to store the database on a network device. The database cannot exist by itself.*

The Examiner then cited In re Mayhew, 527 F. 2d 1229, 188 USPQ 356 (CCPA 1976) and rejected claims 1, 24 and 25 as failing to include elements that are essential to the practice of the invention and, therefore, not enabled.

The Examiner erred in rejecting Appellant's claims 1, 24 and 25 under 25 under 35 U.S.C. 112, first paragraph. Specifically, there is not basis to conclude that the common computing elements cited by the Examiner (i.e., a network, a memory, a network interface card) are essential to the invention. MPEP 2174 states if the specification discloses that a particular feature or element is critical or essential to the practice of the invention, failure to recite or include that particular feature or element in the claims may provide a basis for a rejection based on the ground that those claims are not supported by an enabling disclosure. *See, MPEP 2174, citing In re Mayhew 527 F.2d 1229, 188 USPQ 356 (CCPA 1976).*

The Examiner has provided no evidence indicating that the present specification or statements made by the Appellant disclose that the particular elements identified by the Examiner are critical or essential to the invention. Moreover, the elements cited by the Examiner (e.g., interface cards to receive the claimed message, memory to store the claimed database) are described as examples within the present application. For example, pg. 5, ln. 18-pg. 6, ln. 2 describes a network device 18 as having a network interface generally, of which a network interface card is but one example type.

> *Network interface 18g is configured to enable networking device 18 to communicate with remote client 12 via WAN computer network 16 and with server 14 via LAN computer network 20.* ***An example*** *of a suitable network interface is the Intel Pro/100 card, commercially available from Intel Corporation of Santa Clara, California (emphasis added).*

Similarly, with respect to the recited database, the present application does not disclose that memory per se is a critical or essential element to the claimed invention. For example, the present application on pg 9, ll. 6-8 states:

> *The predefined string* ***may be*** *from a record or a hash table created from previous known headers, and* ***may be*** *stored in memory and/or accessed from a linked database (emphasis added).*

The Examiner has provided no evidence indicating that either the present specification or statements made by the Appellant during prosecution disclose that the particular elements identified by the Examiner are critical or essential to the invention. Furthermore, in contrast to the Examiner's position, the present application describes the elements identified by the Examiner by way of example.

In sum, the elements identified by the Examiner (interface cards, memory) are mundane elements that are not disclosed by Appellant as critical or essential to the invention and, therefore, need not be recited in the claims. The rejection of claims 1, 24, and 25 under 35 U.S.C. 112, first paragraph should be reversed.

The Second Ground of Rejection to be Reviewed on Appeal

In the Office Action, the Examiner rejected claims 1-5, 7, 8, 11-20, 22 and 23 under 35 U.S.C. 103(a) as being unpatentable over a combination of Branstad (USPN 6,842,860) in view of HTTP 1.1, Fielding et al., June 28, 2001, pages 1-6, Chapter 3 Protocol Parameters", pages 1-10, Chapter 4 HTTP Message, pages 1-4 ("Fielding"), Smith et al. (US 6,377,991) and "Official Notice."

Appellant argues claims 1-5, 7, 8, 11-20, 22 and 23 as a group and directs the Board to independent claim 1. The applied references fail to disclose or suggest the inventions defined by Appellant's claim 1, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

As a preliminary comment, with respect to claim 1, the Examiner indicated that, regarding Appellant's usage of the term "wherein," the claim scope is not limited by claim language that suggests or makes optional but does not requires steps to be performed. None of Appellant's claims suggest that the usage of "wherein" renders optional the requirements recited in the "wherein" clauses. Quite the opposite, Appellant's claims clearly recite the "wherein" clauses as requiring specific structural or functional elements. Moreover, the Examiner has not pointed to any specific language in the claims as being construed as optional

Appellant's claim 1 recites a method for comparing an unknown string to a predefined string recites. Claim 1 requires storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within

the database represent known headers for a network communication protocol. Claim 1 further requires receiving, with the network device, a network message, and in response to receiving the network message, selecting one of the plurality of predefined strings stored within database of the network device.

With respect to these elements, the Examiner asserted that Branstad at col. 21, ll. 3-27 and col. 3, ll. 26-38 teach storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol. *Final Office Action, pg. 4*. Branstad does not describe methods and devices for matching strings. Instead, Branstad describes an authentication system that authenticates messages exchanged between devices. *Branstad, Summary*. Branstad at col. 3, ll. 26-38 describes a network device that "generates" an authentication tag 140 for an outbound packet that is being sent by the Branstad device. Brandstad at col. 21, ll. 3-27, also recited by the Examiner, appear to describe functions supported by the Brandstad device for verification of the message by the receiver.

Appellant's claim 1 recites a method that requires storing, on a network device, a plurality of pre-defined strings within a database. Thus, the pre-defined string recited in Appellant's claim 1 must be selected from a database that stores a plurality of pre-defined strings and, therefore, cannot be a portion of the network message received by the device, as suggested by the Examiner on page 3 of the Office Action. Rather, Appellant's claim 1 makes clear that one of the pre-defined strings is selected from the database in response to that device receiving a network message.

To be clear, Branstad's message verification technique cited at col. 3, ll. 26-38 by the Examiner relies on an authentication tag that is generated based on the contents of a network packet. This does not teach or suggest storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol, as required by claim 1. Moreover, this does not teach or suggest in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device, as required by claim 1. Branstad provides no teaching of such features. Rather, the Branstad authentication tag described by the portions of Branstad

cited by the Examiner is dynamically computed based on the particular contents of the packet being sent.

Further, Appellant's claim 1 requires that the predefined strings stored within the database represent "known headers for a network communication protocol." Again, the Examiner relies on Branstad for such teachings, referring to col. 3 at col. 21, ll. 3-27 and col. 3, ll. 26-38. However, these portions of Branstad make clear that the authentication tag (which is compared to a different authentication tag in Branstad) is generated (i.e., computed) from the contents of a network packet using cryptographic techniques.[1] In no manner can the cryptographic authentication tags generated by Branstad based on the content of packets be considered predefined strings that represent known headers for a network communication protocol, as required by claim 1. To the contrary, even in view of the other cited references, any string generated by a cryptographic process simply cannot be a known header for a communication protocol.

Appellant's claim 1 further requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Thus, the literal language of claim 1 limits this claimed embodiment to a method where the predefined string is selected from a database of predefined strings in response to the received message, i.e., after a message is received, and that the unknown string is a portion of that received network message. These features are not taught or suggested by the combination of references cited by the Examiner. Specifically, with respect to these elements the Examiner again refers to Brandstad col. 3, ll. 26-38 that, as stated above, describes the generation of an authentication tag for inclusion within an outbound packet being produced by the sending device; the receiving device in the Branstad system then extracts this authentication tag upon receiving the packet.

Furthermore, Appellant's method of claim 1 for comparing an unknown string to a predefined string further requires performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. In this way, claim 1 literally requires that a bitwise exclusive OR operation be performed between: (1)

[1] Branstad at col. 21, ll. 4-46,

a segment of the unknown string that was identified within a network message, and (2) a segment of the predefined string that was selected from a database of predefined strings in response to the message. Claim 1 also requires the additional step of performing a bitwise operation between a predefined flag and a result of the exclusive OR operation.

As a preliminary comment, the Examiner's rejection of claim 1 did not even refer to or comment on the second step, i.e., performing a bitwise operation between a predefined flag and a result of the exclusive OR operation. See Final Office Action, pg 4. The Examiner failed to even comment on these elements and did not provide any evidence that such elements are known in the art. For at least this reason the rejection must be withdrawn.

Moreover, the Examiner cites Branstad at cols. 19-22 with respect to the step of performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. However, in these columns Branstad describes cryptographic techniques by which the sender generates an authentication tag for an outbound packet. See Branstad, cols. 19-22. Although this process may involve an XOR operation, the cryptographic techniques for generating an authentication tag from a packet fails to teach the requirements of claim 1 of performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string.

In addition, Appellant's claim 1 is limited to performing a bitwise exclusive OR operation to the two strings for which a case-insensitive match is being generated, i.e., between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the predefined string. In other words, the XOR operation of claim 1 is applied to the strings being compared, and those strings being compared are the inputs to that XOR operation. Claim 1, as a whole, requires performing a bitwise operation on a predefined flag and a result of the XOR operation, and comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for the case-insensitive string match. The combination of references cited by the Examiner fails to teach or suggest this process for numerous reasons.

Specifically, none of the references, either singularly or in combination, describe any method that determines whether an unknown string matches a predefined string by performing an XOR operation between the predefined string and the unknown string. As stated in Appellant's response, the Examiner's reasoning is logically flawed in asserting that Branstad teaches performing a bitwise exclusive OR operation between at least a segment of the predefined string and a segment of the unknown string when comparing those two strings. Branstad teaches application of an XOR operation to portions of a message so as to **compute** a binary string, i.e., cryptographic authentication tag. Branstad makes abundantly clear that the XOR operation is not used as part of a comparison operation of the two strings that are provided as inputs to the XOR operation, as required by claim 1. Rather, Branstad describes application of XOR operations only for computing an authentication tag. Only after the tag is generated using cryptographic techniques (which may involve an XOR) is it then compared in a conventional way to a different string, i.e., the authentication tag extracted from the received message.

Col. 3, ll. 35-43 Branstad makes clear that a single authentication tag is computed from an inbound message (by use of cryptographic techniques that may use an XOR as Branstad explains). It is this process that is described in Branstad at col. 19-22 for computing an authentication tag from a packet. Branstad describes a separate comparison operation performed to compare that computed authentication tag to a different authentication tag extracted from the same message:

> Upon receipt of communication **150**, receiver **120** extracts message **130'** and authentication tag **140'**. The extracted message **130'** is used by authentication tag computation module **122** in receiver **120** to produce authentication tag **140"**. A comparison is then made to determine if the generated authentication tag **140"** matches the extracted authentication tag **140'**. If the authentication tags match, then message **130'** is authenticated.

Thus, in no manner does Branstad teach or suggest performing a bitwise XOR operation between a predefined string and an unknown string so that the result of that XOR operation could be used as an indication of a match between those two strings that were applied as inputs to that XOR operation, as required by claim 1.

In other words, the Examiner's argument is based on the premise that different portions of the received message in Branstad could be viewed as two strings, and that Branstad suggests applying an XOR operation to those two strings when computing the cryptographic authentication tag. Even assuming this is a valid interpretation of Branstad, in no way does Branstad in view of the other cited references utilize an XOR operation in a process where the result of the XOR operation could be used to provide an indication that a case-insensitive match exists between those two input strings or the XOR operation, as required by claim 1. For these reasons, Branstad fails to teach or suggest performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string, performing a bitwise operation between a predefined flag and a result of the exclusive OR operation, and comparing the predefined flag and a result of the bitwise operation to produce an indication for a case-insensitive string match, wherein the indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol.

The combination of references cited by the Examiner fail to address this flaw in the Examiner's reasoning with regard to Branstad. Fielding merely describes the HTTP protocol and, in particular, the parameters and headers used when communicating via the HTTP protocol. Fielding does nothing to overcome the fundamental deficiencies of Branstad. Modification of the Branstad authentication technique in view of Fielding, as suggested by the Examiner, would result only in an authentication system in which the messages sent over the network conform to the HTTP protocol. The cryptographic technique used in Branstad to generate the authentication tag from an HTTP message would be the same. To the extent the cryptographic technique (e.g., KR5) of Branstad uses XOR operations and rotations, the operations would be applied to the HTTP message to compute the authentication tag in the system proposed by the Examiner. Fielding provides no suggestion to use an XOR operation for any other purpose, and the combination of Branstad in view of Fielding fails to teach or suggest applying a bitwise

XOR operation between two different messages when determining whether the strings match.

In rejecting claim 1, the Examiner also cites Smith as teaching the "well-known" concept of applying a predefined flag. However, Branstad in view of Fielding and Smith also fail to teach or suggest Appellant's claim 1 for several reasons.

First, at the cited portion, Smith describes "multiplying" the results on an XOR operation by a constant. Branstad in view of Smith does not teach or suggest performing a bitwise operation between a predefined flag and a result of the exclusive OR operation, and then comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for the case-insensitive string match, as required by claim 1.

Second, modification of the Branstad authentication technique in view of Smith makes little or no sense. The result of the XOR operation in Branstad is the generation of the cryptographic authentication tag. In rejecting claim 1 over Branstad in view of Fielding and Smith, the Examiner is proposing to modify the Branstad system so that a predefined flag is applied to the result of the XOR operation, which in Branstad is the authentication tag. The Examiner has failed to explain why one would multiply the authentication tag of Branstad with a constant, as taught by Smith, and how such an operation would in anyway be useful in comparing strings. In fact, the predefined flag (i.e., the "constant") in Smith is used in a hashing function to spread URLs across a hash space. The combination of Branstad in view of Fielding and Smith fails to provide any suggestion as to how a predefined flag could be used with a result of an XOR operation between strings in any manner that would aid detecting a match between two strings.

Third, The Examiner has simply offered no explanation as to how the resultant system could actually achieve a case-insensitive comparison between strings after incorporating such an operation within Branstad. The Appellant is at a loss as to how the modification proposed by the Examiner could even be achieved. The Examiner appears to gloss over the fact that Branstad describes techniques for checking the authentication of a sender by comparing cryptographically generated authentication tags. This cryptographic comparison for purposes of authentication, presumably, would not tolerate matching of characters of different cases, e.g., "a" and "A." It is highly unlikely that a binary digital signature, for example, would be accepted as a match even though it was

different from the expected binary string. Moreover, no actual string comparison function is discussed in Branstad and the comparison function can only be concluded as a conventional string comparison technique.

The Examiner's rejection of claim 1 appears to be nothing more than piecemeal mapping of Appellant's claim language to an aggregation of prior art references combined in a nonfunctional manner. There is no evidence whatsoever that the techniques taught by the references could in any manner be employed to produce a case-insensitive match between two strings, as suggested by the Examiner.

For at least these reasons, modification of Branstad in view of Fielding, Smith and Official Notice fails to establish a prima facie case for non-patentability of Appellant's claims under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

CONCLUSION OF ARGUMENT

The rejections of all pending independent claims should be reversed for the arguments presented with respect to the first grounds of rejection on Appeal.

With respect to the second grounds of rejection on Appeal, Appellant has presented arguments for four different sets of claims under separate headings. To the Extent that the Board disagrees with Appellant's arguments for the first grounds of rejection on Appeal, Appellant respectfully requests separate review of each of the four different sets of claims that have been presented under separate headings with respect to the second grounds of rejection on Appeal.

Respectfully submitted,

Date: September 5, 2008

By: Kent J. Sieffert

Shumaker and Sieffert
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone:(651) 286-8341
Facsimile: (651) 735-1102

Name: Kent J. Sieffert
Reg. No.: 41,312

APPENDIX: CLAIMS ON APPEAL

Claim 1. A computer-implemented method for comparing an unknown string to a predefined string, the method comprising:

storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol;

receiving, with the network device, a network message;

in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device;

identifying a portion of the network message as an unknown string for comparison with the selected predefined string;

performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string;

performing a bitwise operation between a predefined flag and a result of the exclusive OR operation;

comparing the predefined flag and a result of the bitwise operation to produce an indication for a case-insensitive string match, wherein the indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol;

processing the network message based on the indication of the case-insensitive string match; and

outputting a response from the network device based on the processed network message.

Claim 2. The method of claim 1, further comprising identifying a segment of the selected predefined string and identifying a segment of the unknown string for comparison with the identified segment of the selected predefined string.

Claim 3. The method of claim 2, wherein the segment of the selected predefined string and the segment of the unknown string contain a same number of characters.

Claim 4. The method of claim 2, further including left-shifting the binary representation of the segments if the segments contain less than four characters.

Claim 5. The method of claim 2, wherein identifying a case-insensitive string match includes identifying a case-insensitive segment match based on the exclusive OR operation.

Claim 6 (Cancelled).

Claim 7. The method of claim 1, wherein the predefined flag is 0x20202020.

Claim 8. The method of claim 2, further comprising identifying a subsequent segment of the selected predefined string and a subsequent segment of the unknown string for comparison.

Claim 9-10 (Cancelled).

Claim 11. The method of claim 1, wherein the predefined flag is zero.

Claim 12. The method of claim 1, wherein the predefined flag is 0x20.

Claim 13. The method of claim 1, wherein the predefined flag is 0x20202020.

Claim 14. The method of claim 1, wherein the segment of the unknown string and the segment of the selected predefined string each include one character.

Claim 15. The method of claim 1, wherein the segment of the unknown string and the segment of the selected predefined string each include four characters.

Claim 16. The method of claim 1, wherein the unknown string includes an HTTP header field.

Claim 17. The method of claim 1, wherein the selected predefined string is from a table of predetermined HTTP header fields.

Claim 18. The method of claim 1, wherein identifying a case-insensitive match further includes performing another bitwise operation.

Claim 19. The method of claim 1, further comprising identifying the length of the strings.

Claim 20. The method of claim 19, wherein the length of each of the strings are equal.

Claim 21. The method of claim 1, wherein the computer-implemented method is used over a WAN.

Claim 22. The method of claim 1, further comprising determining if characters of the strings are within a predefined ASCII range.

Claim 23. The method of claim 22, wherein characters not within the predefined ASCII range caused the method to yield a negative string match.

Claim 24. A method of case-insensitive string matching for use in a computer network, the method comprising:

storing, on a network device, a plurality of predefined strings, wherein the predefined strings represent known headers for a network communication protocol;

receiving, with the network device, a network message;

selecting one of the plurality of predefined strings stored within the network device;

identifying a portion of the network message as an unknown string for comparison with the selected predefined string;

performing at least one bitwise exclusive OR operation between characters of the selected predefined string and corresponding characters of the unknown string,

performing a bitwise OR operation between a results of the bitwise exclusive OR operation and a predetermined flag; and

comparing the predetermined flag and a result of the bitwise OR operation to produce a single bit output that indicates whether a case-insensitive match exists between the selected predefined string and the unknown string;

processing the network message based on the indication of the case-insensitive match; and

outputting a response from the network device.

Claim 25. A computer networking device for improving data transfer via a computer network, the device comprising a processor configured to compare a client HTTP header with a known HTTP header by:

storing, on the networking device, a database containing a plurality of known HTTP headers;

receiving, with the networking device, a client HTTP header;

in response to receiving the client HTTP header, selecting one of the known HTTP headers stored within the database of the network device;

performing a bitwise exclusive OR operation on binary representations of the client HTTP header and the known HTTP header selected from the database,

performing a bitwise OR operation between a result of the exclusive OR operation and a predetermined flag;

comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for a case-insensitive string match between the client HTTP header and the selected known HTTP header;

processing the network message based on the indication of the case-insensitive match; and

outputting a response from the network device based on the processed network message.

Claim 26. An article of manufacture comprising a storage medium having a plurality of machine-readable instructions, wherein when the instructions are executed by a computing system, the instructions providing for:

storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol;

receiving, with the network device, a network message;

in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device;

identifying a portion of the network message as an unknown string for comparison with the selected predefined string;

performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string;

performing a bitwise operation between a predefined flag and a result of the exclusive OR operation;

comparing the predefined flag and a result of the bitwise OR operation to produce an indication for a case-insensitive string match between the predefined string and the unknown string, wherein the indication for the case-insensitive match indicates whether all characters of the unknown string within the network message match all corresponding characters of the identified predefined string so as to match one of the known headers of the network communication protocol;

processing the network message based on the indication of the case-insensitive match; and

outputting a response from the network device based on the processed network message.

APPENDIX: EVIDENCE

None

APPENDIX: RELATED PROCEEDINGS

None